A cheap feature selection approach for the K-means algorithm

Marco Capó, Aritz Pérez, and Jose A. Lozano

***Abstract*—The increase in the number of features that need to be analyzed in a wide variety of areas, such as genome sequencing, computer vision or sensor networks, represents a challenge for the K-means algorithm. In this regard, different dimensionality reduction approaches for the K-means algorithm have been designed recently, leading to algorithms that have proved to generate competitive clusterings. Unfortunately, most of these techniques tend to have fairly high computational costs and/or might not be easy to parallelize. In this work, we propose a fully-parellelizable feature selection technique intended for the K-means algorithm. The proposal is based on a novel feature relevance measure that is closely related to the K-means error of a given clustering. Given a disjoint partition of the features, the technique consists of obtaining a clustering for each subset of features and selecting the m features with the highest relevance measure. The computational cost of this approach is just $\mathcal{O}(m \cdot \max\{n \cdot K, \log m\})$ per subset of features. We additionally provide a theoretical analysis on the quality of the obtained solution via our proposal, and empirically analyze its performance with respect to well-known feature selection and feature extraction techniques. Such an analysis shows that our proposal consistently obtains results with lower K-means error than all the considered feature selection techniques: Laplacian scores, maximum variance, multi-cluster feature selection and random selection, while also requiring similar or lower computational times than these approaches. Moreover, when compared to feature extraction techniques, such as Random Projections, the proposed approach also shows a noticeable improvement in both error and computational time.**

***Index Terms*—Dimensionality reduction, K-means clustering, feature selection, parallelization, unsupervised learning.**

I. Introduction

Cluster analysis is an essential task for analyzing the ever-growing quantity of high dimensional data from different areas, such as artificial intelligence and pattern recognition [1]–[3]. This technique attempts to partition a data set into disjoint groups called clusters, in such a way that intra-cluster similarity is high and the inter-cluster similarity is low. Perhaps the most well-known clustering algorithm is the so-called K-means algorithm [2], [4]. In fact, it has been identified as one of the top 10 algorithms in data mining [5].

A. K-means problem

Given a set of n data points (instances), $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq \mathbb{R}^d$, and a positive integer K (the number of clusters), the K-means problem consists of splitting the points into K clusters so that the total sum of the squared Euclidean distances of each point to its nearest cluster centroid (center of mass) is minimized. In other words, the goal is to determine a set of K centroids $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$ in $\mathbb{R}^d$ so as to minimize the following quality/error function:

$$E^D(C) = \sum_{\mathbf{x} \in X} \|\mathbf{x} - \mathbf{c}_\mathbf{x}\|_D^2, \tag{1}$$

where $\mathbf{c}_\mathbf{x} = \arg\min_{\mathbf{c} \in C} \|\mathbf{x} - \mathbf{c}\|_D^2$, for all $\mathbf{x} = (x_1, \ldots, x_d) \in X$, and $\|\mathbf{x} - \mathbf{c}\|_S^2 = \sum_{i \in S} (x_i - c_i)^2$, for any $S \subseteq D$, where D is the full set of features of the data set X, i.e., $D = \{1, \ldots, d\}$.

This minimization problem can be seen as a combinatorial optimization problem, since it is equivalent to finding the partition of the n instances in K groups whose associated set of centers of mass minimizes Eq.1 [6]. Unfortunately, finding the solution to such an optimization problem for $K > 1$ is known to be NP-hard [7], even for instances in the plane [8]. For this reason, iterative refinement based algorithms are commonly used to approximate the solution of the K-means problem [9]. Among these methods, the most popular is the K-means algorithm [5], [10].

The K-means algorithm is a heuristic that iteratively relocates the data points between clusters until a locally optimal partition is attained. It consists of two stages: Initialization, in which we set the starting set of centroids, and an iterative stage, known as Lloyd's algorithm.

B. Lloyd's algorithm

Lloyd's algorithm (Algorithm 1) [10] is a two-step recursive heuristic that receives as input an initial set of centroids and determines a local minima of Eq.1. In its first step, each instance is assigned to its closest centroid (assignment step), then the set of centroids is updated as the centers of mass of the instances assigned to the same centroid in the previous step (update step). The time needed for the assignment step is $\mathcal{O}(n \cdot K \cdot d)$, while updating the set of centroids requires $\mathcal{O}(n \cdot d)$ computations. Commonly, the process described above is repeated until the set of centroids remains invariant, in which case such a set of centroids is a local minima of the K-means problem [11].

Among the different advantages, such as the easiness of its implementation, it must be remarked that both phases of Lloyd's algorithm (assignment and update steps) can be easily parallelized [12], which is a major key to meet the scalability of the algorithm [5]. Additionally, there exists a wide variety of speed-ups/approximations to the K-means algorithm, such as different distance prunning approaches [1], [13]–[15], the

M. Capó, A. Pérez and J.A. Lozano are with the Basque Center of Applied Mathematics, Bilbao, Spain, 48009.
E-mail: mcapo@bcamath.org , aperez@bcamath.org
J.A. Lozano is also with the Intelligent Systems Group, Department of Computer Science and Artifitial Intelligence, University of the Basque Country UPV/EHU, San Sebastián, Spain, 20018.
E-mail: ja.lozano@ehu.es

Minibatch K-means [16], the Boundary Weighted K-means [17] and several coreset techniques [18]–[23].

Algorithm 1: Lloyd's algorithm

Input: Data set X, number of clusters K and set of centroids $C = \{\mathbf{c}_1, \dots, \mathbf{c}_K\}$.
Output: Local minima of Eq.1, C^* and clustering $\mathcal{P}$.
while *not Stopping Criterion* **do**
- `Assignment step:`
  - Construct, for all $k \in \{1, \dots, K\}$, the subsets $P_k = \{\mathbf{x} \in X : k = \underset{i \in \{1,\dots,K\}}{\arg\min} \|\mathbf{x} - \mathbf{c}_i\|_D^2\}$.
- `Update step:`
  - Take $\mathbf{c}_k = \overline{P_k}$ for all $k \in \{1, \dots, K\}$.

end
Return $C^* = \{\mathbf{c}_1, \dots, \mathbf{c}_K\}$, $\mathcal{P} = \{P_1, \dots, P_K\}$.

Regarding the disadvantages of Lloyd's algorithm, we could point out its dependence upon the initialization stage in both quality of the obtained solution and running time [24]. A poor initialization, for instance, could lead to an exponential running time in the worst case scenario [25]. A lot of research has been done on this topic: a detailed review of seeding strategies can be found in [26], [27]. In particular, and as widely reported in the literature, at the moment the most popular seeding strategy is the *K-means++* algorithm [28]. This strategy consists of selecting the first centroid uniformly at random from the data set, then each subsequent initial centroid is chosen with a probability proportional to the distance with respect to the previously selected set of centroids. The key idea behind such a seeding technique is to preserve the diversity of seeds while being robust to outliers. The K-means++ algorithm additionally leads to a $\mathcal{O}(\log K)$ factor approximation of the optimal error, on average [28].

On the other hand, the high dimensionality of modern data sets, for instance those related to images, videos, sensor networks, genome sequencing, computer vision and web [29]–[31], has provided a considerable challenge to the K-means algorithm [32], [33]: Not only does the curse of dimensionality make the K-means algorithm slow and affect the usefulness of the Euclidean distance, but the existence of many irrelevant features may not allow the identification of the underlying structure in the data [32], [34]. These facts motivate the study of dimensionality reduction applied to the K-means problem.

C. Dimensionality reduction for the K-means problem

In general, dimensionality reduction is a problem that consists of embedding the original features into a lower dimensional space, which, ideally, contains the same information as the original data set [34]–[36]. In particular, dimensionality reduction can be divided into two main groups, feature extraction and feature selection. In the following sections, we elaborate on both approaches.

1) Feature extraction: In feature extraction, the lower dimensional space is composed of new artificial features that are generated, for instance, via linear combinations of the original features. As can be seen in the literature, both Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) have been succesfully used to preprocess the data prior to executing the K-means algorithm [1], [32], [33], [37]. In particular, among other results, it can be proved that projecting the data set into $m = \lceil \frac{K}{\epsilon} \rceil$ dimensions via SVD allows the clustering error to be preserved within a factor of $1 + \epsilon$ [33]. However, it must be pointed out that the time complexity of both procedures can be fairly high: $\mathcal{O}(\min\{n \cdot d^2, n^2 \cdot d\})$ for SVD and $\mathcal{O}(n \cdot d^2 + d^3)$ for PCA [38]. A fairly cheaper approach consists of applying Random Projections (RP) to the original data set. This technique is just $\mathcal{O}(m \cdot d \cdot n)$ [39]. However, in this case, $m = \mathcal{O}(\frac{K}{\epsilon^2})$ [33] dimensions are required to keep the $(1 + \epsilon)$-approximation factor. Different variants of these methods exist, a detailed review of these can be found in [33], [40].

A different approach that commonly needs much lower computational requirements than the previous dimensionality reduction methods is feature selection. Additionally, this technique tends to improve the interpretability of the obtained clustering when compared to the use of the original set of features [41]–[43].

2) Feature selection: In feature selection, actual dimensions of the data set are selected to construct the lower dimensional space in which the data set is embedded. This approach has been extensively used for the K-means problem [32], [44]. In particular, we can highlight procedures such as the Laplacian scores [45]. This method fundamentally uses a nearest neighbor graph to model the local geometric structure of the data and selects those features which are the smoothest on the graph [45]. Unfortunately, such a construction can be computationally costly as it requires the computation of all pairwise distances between instances. Furthermore, no theoretical guarantees, related to the quality of the obtained clustering, are provided. In the same line, in [44], the Multi-Cluster Feature Selection algorithm (MCFS) is proposed. This method makes use of the eigenvectors of the Laplacian graph, which is defined by the affinity matrix of the data set, to select those features that preserve the multi-cluster structure of the data set the best. Unfortunately, in this case there are also no theoretical guarantees in terms of the quality of the obtained solution and its time complexity is fairly restrictive, $\mathcal{O}(n^2 \cdot d + K \cdot m^3 + n \cdot K \cdot m^2 + d \cdot \log d)$. Another popular unsupervised feature selection consists of selecting those features with the largest variance [44]. The intuition behind this approach is that high variance features may contain more clustering information and, therefore, tend to have a higher impact on the error function, Eq.1 [45].

Other feature selection approaches consist of selecting features via random sampling, such as in [46], [47] (uniformly) or [32], where a more elaborated SVD-based selection procedure is proposed. For this last procedure, it can be proved that selecting $m = \Theta(\frac{K \cdot \log \frac{K}{\epsilon}}{\epsilon^2})$ dimensions and running any λ-approximate K-means algorithm[1] leads to a $(1 + (1 + \epsilon) \cdot \lambda)$-

[1]Given $\lambda \geq 1$, algorithm $\mathcal{A}$ is a λ-approximate K-means algorithm if it can converge to a set of centroids, C', satisfying $E^D(C') \leq \lambda \cdot E^D(C_{opt})$, where $C_{opt} = \underset{C \subseteq \mathbb{R}^d, |C|=K}{\arg\min} E^D(C)$. For instance, if algorithm $\mathcal{A}$ is taken as the K-means algorithm, then $\lambda = 1$. However, we could also consider algorithm $\mathcal{A}$ as any variant of the K-means algorithm, such as those mentioned in Section I-B.

approximation of the optimal K-means error, with high probability. Unfortunately, this technique may not result as being very practical, as it is $\mathcal{O}(\min\{n \cdot d^2, n^2 \cdot d\})$.

D. Contribution

In this work, we propose a fully parallelizable feature selection algorithm for the K-means problem called the *Relevance Based Feature Selection for the K-means* algorithm (KMR). In the first step of KMR, the set of features is partitioned into multiple subsets and a clustering for the entire data set, restricted to each set of features, is learnt in parallel. Taking into consideration each of the obtained clusterings, we quantify the importance (relevance) of each feature and select those that seem to have a greater influence when determining the corresponding clustering. For this purpose, we first define a measure to compute a feature relevance for a given clustering, which is presented in Section II and comment on some of its properties. Afterwards, in Section III, we elaborate on the KMR algorithm and provide some theoretical guarantees for the obtained approximation. In Section IV, we analyze the performance of the proposed algorithm with respect to different feature selection/extraction techniques in terms of the clustering quality and the computational time. Finally, in Section V, we define the next steps and possible improvements to our current work.

II. A Feature Relevance Measure for the K-means Clustering Problem

In this section, we define a measure that allows us to quantify the importance of a given variable with respect to a predefined K-means clustering. In particular, given a partition of the dataset, the proposed measure consists of evaluating the effect of fixing a variable on the obtained clustering in terms of the associated K-means error increase.

In order to provide some intuition, in Fig. 1, we show an example on a 2D mixture of Gaussians and 2 clusters. In Fig. 1(a), in different colors, a clustering, $\mathcal{P}$, and its associated centers of mass (black dots), C, are presented. Moreover, in Fig. 1(b) and Fig. 1(c), we can observe the variation on the clusterings that takes place when the centers of mass, in C, are fixed to a given value in either dimension. In this case, it is clear that dimension 2 (y-axis) provides less information of the obtained clustering structure, since, when fixed, the clustering remains invariant with respect to the original one. Therefore, dimension 1 (x-axis) is a better candidate to be selected. Rather than analyzing the clustering variations, in practice, we will evaluate the error increase with respect to the original clustering when a certain feature is removed[2]. Then, we will select the subset of features that produce the largest error increase.

[2]We say that a feature $s \in D \setminus S$ is removed when, for each centroid, we fix the entry associated to that variable to its corresponding center of mass on X, $\overline{X^{\{s\}}}$.

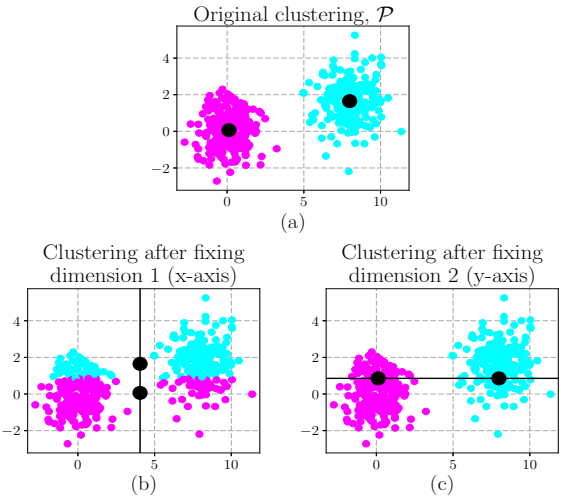

Fig. 1. Illustration of the proposed feature selecion rule.

Next, we introduce the notation used throughout this work. From now on, the data set X restricted to a subset of features, $S \subseteq D = \{1, ..., d\}$, is denoted by X^S. $D_{i:j}$ is the subset of D containing all the variables from the i^{th} feature to the j^{th} feature of D. Furthermore, by $\mathcal{P} = \{P_1, \dots, P_K\}$ we refer to a given partition/clustering of X into K groups and by $C = \{\mathbf{c}_1, \dots, \mathbf{c}_K\}$ to the set of centroids associated to it. That is, $\mathbf{c}_k = (c_{k,1}, \dots, c_{k,d}) = \overline{P_k}$, for $k = \{1, \dots, K\}$. Finally, given $S \subseteq D$ and $C = \{\mathbf{c}_1, \dots, \mathbf{c}_K\}$, we denote by $C^S = \{\mathbf{c}_1^S, \dots, \mathbf{c}_K^S\}$ to the set of centroids satisfying $c_{k,s}^S = c_{k,s}$, for $s \in S$, and $c_{k,s}^S = \overline{X^{\{s\}}}$, for $s \in D \setminus S$ and $k \in \{1, \dots, K\}$. For instance, $C^{\{1\}}$ and $C^{\{2\}}$ are the set of centroids composed of the black dots in Fig. 1(c) and Fig. 1(b), respectively. Following the intuitions illustrated in Fig. 1, in Definition 1, we provide a relevance measure of a feature for a given clustering, $\mathcal{P}$:

Definition 1. *Given a data set X with features D, a clustering of X, $\mathcal{P}$, and its associated set of centroids, C, we define the relevance of a feature $s \in D$, for $\mathcal{P}$, as follows:*

$$r_{\{s\}} = \sum_{k=1}^{K} |P_k| \cdot (c_{k,s} - \overline{X^{\{s\}}})^2 \tag{2}$$

One of the main advantages of such a relevance measure is the fact that it is additively decomposable, i.e., the relevance of a set of features $S \subseteq D$, for a clustering $\mathcal{P}$, can be written as $r_S = \sum_{s \in S} r_{\{s\}}$. This characteristic will allow us to progressively select those features with the largest impact on the K-means clustering without evaluating the K-means error gain for all the possible combinations of variables to be selected, which would be quite costly ($\mathcal{O}(\binom{d}{m} \cdot n \cdot K \cdot d)$ time cost). Furthermore, the following result shows that the relevance can be interpreted in terms of the increment of the K-means error function when the value of a feature is fixed to its corresponding center of mass:

Theorem 1. *Given a data set X with features D, a clustering $\mathcal{P} = \{P_1, \dots, P_K\}$ and its associated set of centroids, C, then,*

for any subset of features $S \subseteq D$, the following inequality holds:

$$E^D(C^S) \leq E^D(C) + r_{D \setminus S} \tag{3}$$

The result of Theorem 1 shows that the proposed relevance measure can be used to upper-bound the increment of the K-means error function when a subset of dimensions is fixed. It should be noted that computing the scores of all the variables $s \in D$ can be done in just $\mathcal{O}(K \cdot |D|)$ time. Moreover, since the relevance is additively decomposable, the effect of the removal of any subset of variables in the K-means error function is also additive (Eq.2 and Eq.3). Therefore, by computing the relevance of each feature individually, it is possible to upper-bound the error increment that is produced by the removal of each subset of features. This interesting property allows us to construct a $(1+\varepsilon)$-approximation of $E^D(C)$ by selecting a subset of features according to the proposed relevance measure.

Corollary 1. *Given a data set X with features D, a clustering $\mathcal{P}$, its associated set of centroids C, and a subset of features $S \subseteq D$, then C^S is a $(1+\epsilon)$-approximation of $E^D(C)$, where*

$$\epsilon \leq \epsilon_D^S = \frac{r_{D \setminus S}}{E^D(C)} \tag{4}$$

Note that $\epsilon_D^S \geq 0$ is given in terms of the relevance of the removed features, $D \setminus S$. This result motivates the following feature selection algorithm (Algorithm 2) which minimizes the number of features selected such that the obtained clustering, on those variables, S, is a $(1+\epsilon)$-approximation of the K-means error function evaluated on the full set of features, D.

Algorithm 2: Feature selection based on Corollary 1

Input: Data set $X \in \mathbb{R}^{n \times d}$, clustering $\mathcal{P}$ and its associated set of centroids, C.
Output: Set of features S, for which C^S a $(1+\epsilon)$-approximation of $E^D(C)$, where $D = \{1, \ldots, d\}$.
- Compute the relevance of each feature $s \in D$, $r_{\{s\}}$, for $\mathcal{P}$.
- Sort D increasingly according to $\{r_{\{s\}}\}_{s \in D}$.
- Determine the largest index k s.t. $r_{D_{1:k}} \leq \varepsilon \cdot E^D(C)$ holds.
Return $S = D_{k+1:|D|}$.

It should be noted that, for the output of Algorithm 2, S, C^S is a $(1+\epsilon_D^S)$-approximation of C, where $\epsilon_D^S \leq \epsilon$[3], while the computational cost of generating such an approximation is just $\mathcal{O}(|D| \cdot \max\{K, \log|D|\})$. Furthermore, we would like to point out that, in the following sections, we will refer by ξ_D^m to the value of $\epsilon > 0$ obtained when selecting m features of D via Algorithm 2, i.e., $\xi_D^m = \epsilon_D^S$, where $m = |D| - k$.

Algorithm 2 allows us to select a set of features for which we can preserve a predefined error quality guarantee, however, for this approach, we can not fix the cardinality of such a set in advance, as it occurs for the feature selection/extraction algorithms. In the following section, we discuss a distributed feature selection algorithm that allows us to fix the number of features selected and which is partially related to Algorithm 2.

[3]In Appendix B, we briefly present some additional experimental results showing the accuracy of the error bound obtained via Algorithm 2.

III. Distributed feature selection algorithm for K-means: KMR

One of the main motivations of this work is to improve the scalability of Lloyd's algorithm with respect to the dimensionality of the K-means problem. Moreover, in the literature there are different competitive approximations to the K-means algorithm, such as [17], [20], [22], [48], that do not scale well with respect to this factor. The proposed feature selection algorithm will allow the use of these techniques to approximate the solution of the clustering problem on a tractable number of dimensions for them.

In this section, we formally introduce the Relevance Based Feature Selection for the K-means algorithm (KMR). KMR is a distributed algorithm that performs feature subset selection by using the relevance measure given in Definition 1. Initially, KMR constructs a partition of the dimensions set and obtains a clustering for each subset of dimensions by using an approximate algorithm $\mathcal{A}$ for the K-means problem, e.g., K-means algorithm or the methods proposed in [17], [20], [22], [48]. Next, it quantifies the relevance of each feature for its corresponding clustering and selects the most relevant features. This simple and distributed algorithm has different theoretical guarantees in terms of the K-means error function (see Section III-A). The pseudocode of KMR is presented in Algorithm 3.

Algorithm 3: Relevance Based Feature Selection for the K-means (KMR)

Input: Data set $X \in \mathbb{R}^{n \times d}$, number of clusters K, number of features to be selected $m < d$ and a heuristic for the K-means problem, algorithm $\mathcal{A}$.
Output: A subset of features of size m, S.
- Generate a partition of $D = \{1, \ldots, d\}$, $\{D_1, \ldots, D_t\}$.
for $i = 1, \ldots, t$ **do**
  - Obtain a clustering $\mathcal{P}_i$ of X^{D_i} using algorithm $\mathcal{A}$.
  - Compute the relevance of each feature $s \in D_i$, $r_{\{s\}}$, for $\mathcal{P}_i$.
end
Return $S = \bigcup_{i=1}^{t} S_i^*$ with $|S| = m$, where

$$\{S_1^*, \ldots, S_t^*\} = \underset{S_i \subseteq D_i \text{ for } i \in \{1,..,t\}}{\arg\min} \max\{\epsilon_{D_1}^{S_1}, \ldots, \epsilon_{D_t}^{S_t}\} \tag{5}$$

The first task in KMR (Algorithm 3) is to partition the set of features D into the smallest number of chunks possible, t, so that all parties have a similar amount of features, which are upper-bounded by m. The goal behind this step is to make sure that algorithm $\mathcal{A}$ is applied the lowest number of times over a number of dimensions that is tractable for it. Observe that, for the first requirement, the lowest number of parties needed is given by $t = \lceil \frac{d}{m} \rceil$. Moreover, for the latter

requirement, it is always possible to find a partition of D, for which $\max_{i\in\{1,\dots,t\}} |D_i| - \min_{i\in\{1,\dots,t\}} |D_i| \leq 1$[4]. Afterwards, algorithm $\mathcal{A}$ is distributedly applied on each chunk D_i, for $i = 1,\dots,t$, to determine the clusterings required to compute the relevance of each feature in D_i: If we set algorithm $\mathcal{A}$ to be the standard K-means algorithm [10], K-means++ algorithm [28] or even the Boundary Weighted K-means algorithm [17], its computational complexity is bounded by $\mathcal{O}(n \cdot K \cdot m)$ and, computing the relevance of each feature in D_i, is just $\mathcal{O}(K \cdot m)$.

The last step of KMR is to select the features set, $S = \bigcup_{i=1}^{t} S_i^*$, using the computed relevances in parallel in Algorithm 3. To do so, we need to compute, in a similar fashion to Algorithm 2, the list $\{\xi_{D_i}^{|D_i|}, \xi_{D_i}^{|D_i|-1}, \dots, \xi_{D_i}^{0}\}$ for each chunk of dimensions D_i. This step can be done in just $\mathcal{O}(|D_i| \cdot \max\{K, \log|D_i|\})$ time, for all $i \in \{1,\dots,t\}$. Afterwards, $\{S_1^*,\dots,S_t^*\}$ is determined via Eq.5. This last phase can be done in $\mathcal{O}(t)$, as shown in Appendix C.

Following the previous analysis, the computational cost of KMR is, at most, $\mathcal{O}(\max\{m \cdot \max\{n \cdot K, \log m\}, t\})$ on each subset. That is, if the following unrestrictive constraints are satisfied $d \leq n \cdot K \cdot m^2$ and $m \leq 2^{n \cdot K}$, then the cost of KMR is dominated by the time complexity of algorithm $\mathcal{A}$, i.e., $\mathcal{O}(n \cdot K \cdot m)$. A further interesting remark is that KMR, in the extreme case $K = n$, is equivalent to selecting the m features with the largest variances, which is another commonly used feature selection strategy [44], [45].

A. Quality guarantees of KMR

Once m features, $S \subseteq D$, are selected via Algorithm 3, algorithm $\mathcal{A}$ can be applied over the data set X^S to approximate the clustering with a reduced number of dimensions. This last step has $\mathcal{O}(n \cdot K \cdot m)$ time complexity. In this section, we provide different theoretical guarantees for the quality of the obtained approximation using the features selected via KMR (Algorithm 3).

We start by analyzing and providing a theoretical quality for approximations that are obtained via Algorithm 2: instead of choosing the number of the selected features, m, in advance, we consider an alternative parameter, $\epsilon > 0$, that imposes a threshold to the cumulative relevance of the selected features normalized by the error of the associated clustering.

Theorem 2. *Given a data set X with features D, a constant $\epsilon > 0$ and a partition of D, $\{D_1,\dots,D_t\}$, if S_i is the set of features selected via Algorithm 2 from D_i, for $\epsilon > 0$, then the output of a λ-approximate K-means algorithm (algorithm $\mathcal{A}$) on $S = \bigcup_{i=1}^{t} S_i$, C^*, satisfies*

$$E^D(C^*) \leq \varphi \cdot (1+\epsilon) \cdot E^D(C_{opt}) - \delta \qquad (6)$$

where $C_{opt} = \operatorname{arg\,min}_{C\subseteq\mathbb{R}^d, |C|=K} E^D(C)$, $\varphi = \lambda^2 \cdot \frac{\Theta_S}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)}$ *with* $\Theta_S = E^S(\{\overline{X^S}\})$ *and* $\delta = \frac{\lambda\cdot(1+c)-t}{t} \cdot \Theta_{D\setminus S}$ *with* $c = \frac{\sum_{i=1}^{t} E_{D_i}^{S\setminus S_i}(C_i)}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)}$.

We must point out that, in Eq.6, the value of δ is commonly non-negative [5]. For this reason, Theorem 2 shows that applying algorithm $\mathcal{A}$ using the selected features by Algorithm 3 tends to lead to a $\mathcal{O}(1+\epsilon)$-approximation of the K-means problem over the whole data set, X. In other words, performing the proposed feature selection procedure, which locally controls the increase of the K-means error in different groups of dimensions, allows us to preserve the error bounds over the entire set of dimensions. Unfortunately, the number of selected features depends on the values of the normalized relevances of the features and on the value of $\epsilon > 0$. Often, in practice, we want to fix the number of selected features. To tackle this difficulty, we can consider different values $\epsilon > 0$, for each party D_i in $i \in \{1,\dots,t\}$, such that the total number of selected features is $|S| = m$. In this regard, we can take into consideration the following corollary.

Corollary 2. *If $S = \bigcup_{i=1}^{t} S_i^*$ is the set of m features selected by Algorithm 3 on a data set X with features D, then the output of a λ-approximate K-means algorithm (algorithm $\mathcal{A}$) on S, C^*, satisfies*

$$E^D(C^*) \leq \varphi \cdot (1+\epsilon^*) \cdot E^D(C_{opt}) - \delta, \qquad (7)$$

where $\epsilon^* = \max_{i\in\{1,\dots,t\}} \epsilon_{D_i}^{S_i^*}$.

Corollary 2 shows the importance of minimizing the value $\epsilon > 0$ used on each chunk of dimensions D_i needed to select exactly m variables overall, which is the last step of Algorithm 3. Using the arrays generated by Algorithm 2, $\{\xi_{D_i}^{|D_i|}, \xi_{D_i}^{|D_i|-1}, \dots, \xi_{D_i}^{0}\}$ for all $i \in \{1,\dots,t\}$, this problem can be naively solved in $\mathcal{O}(d \cdot \log m)$ time. Furthermore, since all these arrays are sorted, the optimization problem in the last step of Algorithm 3 can be easily solved in $\mathcal{O}(t)$ time, see Appendix C for more details.

IV. Experiments

In this section, we perform a series of experiments so as to analyze the trade-off between the computational time and the quality of the approximation obtained by the K-means algorithm after applying it on a wide variety of preprocessed data sets via different dimensionality reduction techniques.

Given a predefined number of features to be extracted/selected, we compare the performance of the K-means Relevance Feature Subset Selection (K**MR**) with respect to the K-means algorithm[6] applied on m features selected via i) Laplacian Scores (**LS**), ii) Maximum variance (**MaxVar**), iii) Multi-Cluster Feature Selection (**MCFS**) and iv) Uniformly at

[4]For instance, if we set $f = m \cdot t - d$, then we can take $f - \lfloor \frac{f}{t} \rfloor \cdot t$ parties, with $m - \lfloor \frac{f}{t} \rfloor - 1$ dimensions, and the remaining $t - f + \lfloor \frac{f}{t} \rfloor \cdot t$ parties, with $m - \lfloor \frac{f}{t} \rfloor$ dimensions.

[5]See Observation 1 in Appendix A.

[6]In order to guarantee the obtained clustering to be competitive, we use K-means algorithm initialized via K-means++ (K**M++**).

random (**Rand**), or extracted via i) Random Projections (**RP**), ii) Principal Component Analysis (**PCA**) and iii) Singular Value Decomposition (**SVD**). We analyze the performance of these methods on a wide variety of well-known real data sets (see Table I) with different scenarios of the clustering problem. The number of features to be selected/extracted is fixed as $m \in \{10, 25, 50, 75, 100\}$[7]. Furthermore, due to the random nature of the experimental setting, each experiment has been repeated 20 times.

For each experimental setting, we evaluate the relative K-means error obtained for each method M, $\hat{E}_M = \frac{E_M - E_{KM++}}{E_{KM++}}$, where E_M and E_{KM++} stand for the K-means error of method $M \in \{$ KMR, LS, MaxVar, MCFS, Rand, PCA, RP, SVD$\}$ and the error obtained by KM++ over all the dimensions of the data set, respectively. To analyze how well each method preserves the clustering quality when compared to KM++, we additionally present the Adjusted Rand Index, ARI [49], [50], of each method M with respect to the clustering obtained by KM++. In terms of the computational resources, we show the proportion of the computational time of each method M, t_M, with respect to that of KM++, $\hat{t}_M = \frac{t_M}{t_{KM++}}$. To start the analysis, in Tables II-III, we show the average for these factors, over all the data sets and considered methods.

Table II: **Relative error** - average over all data sets-.

Method	m=10	m=25	m=50	m=75	m=100
KMR	4.1×10^{-2}	1.2×10^{-2}	6.4×10^{-3}	4.0×10^{-3}	2.3×10^{-3}
LS	2.7×10^{-1}	9.8×10^{-2}	7.6×10^{-2}	4.8×10^{-2}	5.0×10^{-2}
MaxVar	8.6×10^{-2}	2.5×10^{-2}	1.3×10^{-2}	1.0×10^{-2}	9.0×10^{-3}
MCFS	1.7×10^{-1}	1.3×10^{-1}	1.1×10^{-2}	8.1×10^{-2}	6.1×10^{-2}
Rand	2.9×10^{-1}	2.0×10^{-1}	1.6×10^{-1}	1.2×10^{-1}	1.2×10^{-1}
PCA	1.6×10^{-3}	1.7×10^{-3}	8.4×10^{-6}	5.3×10^{-5}	7.7×10^{-5}
RP	9.3×10^{-2}	3.5×10^{-2}	1.2×10^{-2}	1.3×10^{-2}	7.8×10^{-3}
SVD	4.0×10^{-4}	5.9×10^{-4}	7.2×10^{-5}	8.4×10^{-5}	7.9×10^{-5}

Table III: **(ARI, Relative computational time)** - average over all data sets-.

Method	m=10	m=25	m=50	m=75	m=100
KMR	(0.69, 0.28)	(0.75, 0.34)	(0.77, 0.27)	(0.80, 0.25)	(0.83, 0.20)
LS	(0.42, 5.78)	(0.60, 5.82)	(0.56, 1.56)	(0.63, 1.60)	(0.60, 1.51)
MaxVar	(0.63, 0.27)	(0.72, 0.32)	(0.70, 0.25)	(0.71, 0.23)	(0.68, 0.19)
MCFS	(0.34, 2.21)	(0.50, 3.16)	(0.40, 2.08)	(0.39, 2.35)	(0.48, 2.85)
Rand	(0.23, 0.29)	(0.37, 0.33)	(0.27, 0.27)	(0.45, 0.25)	(0.46, 0.20)
PCA	(0.93, 0.42)	(0.95, 0.54)	(0.94, 0.52)	(0.95, 0.53)	(0.95, 0.55)
RP	(0.49, 0.31)	(0.63, 0.35)	(0.66, 0.29)	(0.68, 0.28)	(0.74, 0.23)
SVD	(0.92, 0.40)	(0.94, 0.50)	(0.94, 0.50)	(0.95, 0.54)	(0.96, 0.60)

At first glance, it can be seen that, among the feature selection techniques (LS, MaxVar, MCFS, Rand), KMR obtains on average both the lowest relative error and largest ARI with respect to the clusterings achieved by KM++. In particular, for the different numbers of features to be selected, m, KMR consistently obtains an average error with a relative error under 0.05 with respect to KM++. That is, the clustering obtained after executing K-means++ on the features selected by KMR usually had an error increment of under 5% of the lowest error achieved by KM++ over the original data set. On the other hand, MaxVar also generated competitive approximations, however, for the largest numbers of features selected $m = \{50, 75, 100\}$, it has 1 order of magnitude of additional error when compared to KMR, as well as a significantly smaller ARI (for $m = 100$, under 0.15 smaller than KMR). Moreover, LS, MCFS, and Rand obtained the largest relative errors (for all settings, at least, one order of magnitued larger than KMR) and the smallest ARI (under 0.15 with respect to KMR, in every case). For the feature extraction techniques, it can be seen that that SVD and PCA obtained fairly competitive approximations, preserving the clustering structure achieved by KM++ almost identically, while RP generates less accurate approximations, which are commonly improved by KMR. It is clear that, as we increase the number features to be extracted/selected, the quality of the approximations, for the different methods, tends to improve dramatically. On the other hand, in Table III, we observe that all the feature selection approaches, except for LS and MCFS, have similar computational times, which are, on average, under 35% of the time required by KM++. For the feature extraction methods, we can see that PCA and SVD required, on average, up to 3 times the computational time needed by KMR when extracting the largest number of variables.

A. Feature Selection

In Fig. 2, we can observe the results obtained in all 16 data sets for feature selection. As previously mentioned, among these methods, KMR obtained the most accurate approximations, reducing, on average, at least one order of relative error for $m \in \{50, 75, 100\}$ and regularly reaching an ARI w.r.t. KM++ clustering over 0.05 higher than that achieved by the other approaches. On the downside, LS, MCFS, and Rand consistently obtained the least competitive clusterings by far.

In spite of the competitive performance in terms of the quality of the approximation, KMR required the same order of computational time as MaxVar and Rand. Such behavior is expected, as all these three methods have a time complexity dominated by the K-means run, over the selected variables, which is linear w.r.t. n, K and m. To observe this in more detail and, as the data sets presented in Table I have

[7]In order to avoid the performed dimensionality reduction to be negligible, for those data sets with $d \leq 100$, we set $m \leq \frac{3}{4} \cdot d$, e.g., for mfeat Fourier ($d = 76$), we analyze reductions to $m \in \{10, 25, 50\}$ features.

Table I: Information of the data sets.

Data Set	n	K	d
KC1 Binary	145	2	91
Amazon Mechanical	180	10	500
Micro Mass	571	20	1300
Breast Cancer	604	2	10936
Arcene NIPS	700	2	10000
Gene RNA-Seq	801	5	20531
Ova Uterus	1545	2	10936
Madelon NIPS	2000	2	500
mfeat Fourier	2000	10	76
Scene	2407	3	299
GINA Agnostic	3468	2	970
Bio Response	3751	2	1776
Spambase	4601	2	57
Waveform Generator	5000	3	40
Satellite Image	6430	6	36
USPS LeCun	7291	10	256

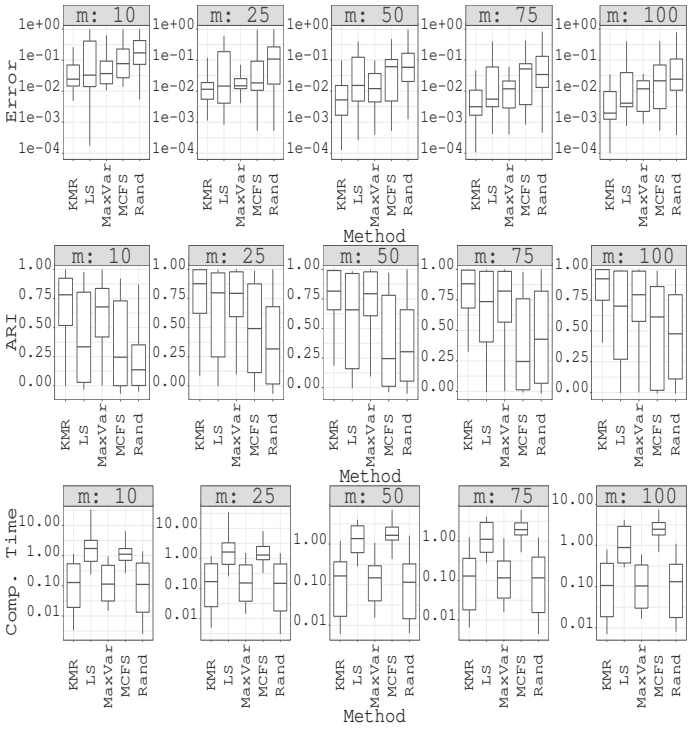

Fig. 2. Feature selection output for all data sets -boxplot-.

quite different characteristics, we have divided each factor (dimensionality, number of instances and classes) into three regularly-sized groups and computed the relative error, relative computational time and ARI for all the feature selection methods, see Tables IV-VI.

Table IV: **(Relative error, ARI, Relative computational time)** - average over groups of dimensions-.

METHOD	$d \leq 100$	$100 < d < 10000$	$d \geq 10000$
KMR	$(1.1 \times 10^{-2}, 0.88, 0.80)$	$(2.1 \times 10^{-2}, 0.68, 0.23)$	$(7.0 \times 10^{-3}, 0.91, 0.01)$
LS	$(2.6 \times 10^{-1}, 0.59, 12.47)$	$(6.8 \times 10^{-2}, 0.54, 2.01)$	$(1.2 \times 10^{-1}, 0.53, 0.54)$
MAXVAR	$(6.4 \times 10^{-2}, 0.81, 0.77)$	$(3.1 \times 10^{-2}, 0.52, 0.22)$	$(1.3 \times 10^{-2}, 0.84, 0.03)$
MCFS	$(2.9 \times 10^{-1}, 0.53, 5.66)$	$(4.3 \times 10^{-2}, 0.45, 1.61)$	$(1.1 \times 10^{-1}, 0.35, 2.07)$
RAND	$(2.3 \times 10^{-1}, 0.57, 0.78)$	$(1.8 \times 10^{-1}, 0.31, 0.25)$	$(1.8 \times 10^{-1}, 0.33, 0.01)$

Table V: **(Relative error, ARI, Relative computational time)** - average over groups of number of clusters-.

METHOD	$K \leq 3$	$3 < K < 10$	$K \geq 10$
KMR	$(5.1 \times 10^{-3}, 0.75, 0.18)$	$(1.6 \times 10^{-2}, 0.91, 0.40)$	$(3.0 \times 10^{-2}, 0.70, 0.41)$
LS	$(1.3 \times 10^{-1}, 0.52, 2.82)$	$(8.3 \times 10^{-2}, 0.63, 7.16)$	$(1.1 \times 10^{-1}, 0.60, 2.04)$
MAXVAR	$(1.1 \times 10^{-2}, 0.65, 0.18)$	$(6.5 \times 10^{-2}, 0.84, 0.38)$	$(5.4 \times 10^{-2}, 0.66, 0.37)$
MCFS	$(1.6 \times 10^{-1}, 0.27, 2.98)$	$(3.8 \times 10^{-2}, 0.60, 2.62)$	$(8.3 \times 10^{-2}, 0.69, 1.11)$
RAND	$(1.4 \times 10^{-1}, 0.28, 0.17)$	$(1.4 \times 10^{-1}, 0.56, 0.45)$	$(3.5 \times 10^{-1}, 0.44, 0.39)$

Table VI: **(Relative error, ARI, Relative computational time)** - average over groups of number of instances-.

METHOD	$n \leq 750$	$750 < n < 2500$	$n \geq 2500$
KMR	$(1.1 \times 10^{-2}, 0.81, 0.28)$	$(5.0 \times 10^{-3}, 0.89, 0.24)$	$(3.0 \times 10^{-2}, 0.62, 0.31)$
LS	$(1.7 \times 10^{-1}, 0.42, 0.66)$	$(8.6 \times 10^{-2}, 0.70, 1.75)$	$(1.0 \times 10^{-1}, 0.60, 8.79)$
MAXVAR	$(1.8 \times 10^{-2}, 0.79, 0.26)$	$(5.1 \times 10^{-3}, 0.85, 0.23)$	$(7.7 \times 10^{-2}, 0.40, 0.30)$
MCFS	$(1.3 \times 10^{-1}, 0.30, 2.17)$	$(9.1 \times 10^{-2}, 0.63, 1.98)$	$(1.3 \times 10^{-1}, 0.32, 3.56)$
RAND	$(3.0 \times 10^{-1}, 0.31, 0.32)$	$(1.4 \times 10^{-1}, 0.37, 0.23)$	$(1.0 \times 10^{-1}, 0.42, 0.28)$

According to Tables IV-VI, it is clear that, regardless of the clustering scenario, KMR, on average, provides both the lowest relative error and largest ARI w.r.t. KM++. However, in spite of the groups selected for each factor (dimensionality, number of instances and classes), we do not notice a major variation in terms of the quality of the obtained clusterings. As shown, in both Tables II-III and Fig. 2, the number of features selected indeed has a larger effect on the comparison of the clusterings obtained by the different methods considered w.r.t. KM++. As previosuly discussed, in terms of the computational time, for the different configurations considered, KMR, MaxVar, and Rand have comparable time requirements, however LS and MCFS perform quite poorly in this regard as well, especially as the number of instances is increased, e.g., for the group "$n \geq 2500$", LS and MCFS required 28.35 and 11.48 times the computational running time of KMR, on average, respectively.

B. Feature Extraction

In this section we perform the same analysis as in Section IV-A, however this time comparing KMR to the different feature extraction techniques considered: In Fig. 3, we observe the results obtained for all the data sets w.r.t. the number of features extracted. Moreover, in Tables VII-IX, we present the relative error, relative computational time and ARI and for the different groups of dimensionality, number of instances and classes.

In this case, the analysis of the results, especially in terms of the computational time, is more interesting than in Section IV-A, as the time demands are not neccesarily dominated by the corresponding K-means run for each method. We can see in Tables VII-IX that there is no clear improvement/ diminishment, in terms of clustering quality (relative error and ARI), as we increase any of the three factors. As we show in Tables II-III and Fig. 3, the quality of the obtained clustering seems to be largely dominated by the number of features extracted, regardless of the method, as in Section IV-A. In any case, despite the considered setting, we must remark that KMR outperforms the clustering quality of RP, in both ARI and error.

In terms of the computational time required w.r.t. KM++, we observe a large effect of the orginal dimensionality of the data set, d. Unfortunately, as the dimensionality reduction step via SVD and PCA is supralinear w.r.t. d, this phase is far more time consuming than running the K-means algorithm on the extracted/selected variables. For this reason, we can see that in the group "$d \leq 100$", PCA and SVD just need, on average, 1.17 and 1.16 times the computational time of KMR, while, for "$d \geq 10000$", the time demands increase to 29.16 and 30.08 times, respectively. In other words, methods such as KMR and RP can be more suitable for preprocessing massive data sets.

On the other hand, KMR also outperforms RP time-wise. However, in this case the time reduction is not that relevant. This is due to the fact that the computational time of RP also grows linearly with respect to the different factors, and that its feature extraction step is independent of the number of classes. Hence, as we consider larger values of clusters, we expect a more competitive perfomance in terms of computational resources for RP, especially when $d \leq K$, which is a very unlikely case as most of the applications considered for this analysis tend to have massive dimensionalities.

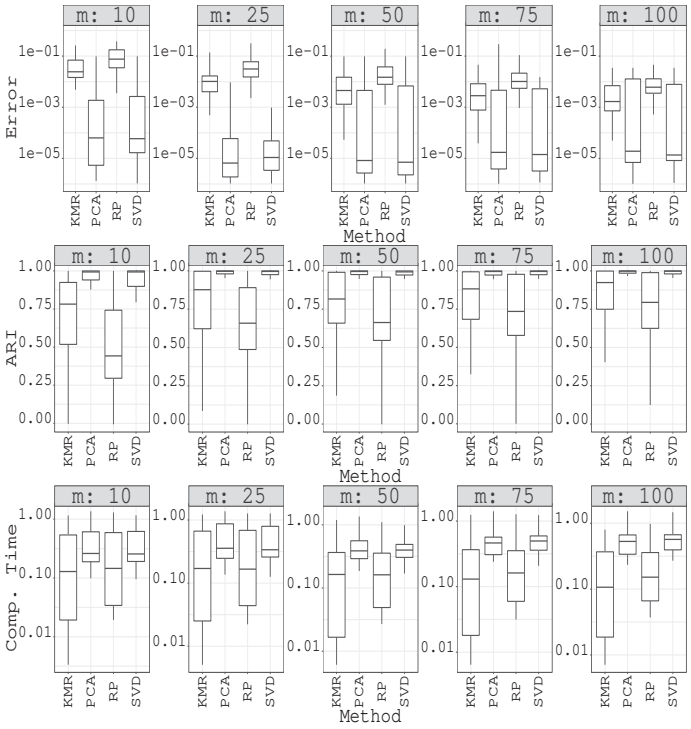

Fig. 3. Feature extraction output for all data sets -boxplot-.

Table VII: **(Relative error, ARI, Relative computational time)** - average over groups of dimensions-.

METHOD	$d \leq 100$	$100 < d < 10000$	$d \geq 10000$
KMR	$(1.1 \times 10^{-2}, 0.88, 0.80)$	$(2.1 \times 10^{-2}, 0.68, 0.23)$	$(7.0 \times 10^{-3}, 0.91, 0.01)$
PCA	$(4.0 \times 10^{-3}, 0.94, 0.90)$	$(3.4 \times 10^{-5}, 0.93, 0.49)$	$(2.2 \times 10^{-5}, 0.97, 0.35)$
RP	$(5.1 \times 10^{-2}, 0.73, 0.79)$	$(3.5 \times 10^{-2}, 0.53, 0.27)$	$(3.7 \times 10^{-2}, 0.73, 0.04)$
SVD	$(1.4 \times 10^{-3}, 0.93, 0.91)$	$(1.1 \times 10^{-5}, 0.92, 0.43)$	$(3.0 \times 10^{-5}, 0.98, 0.36)$

Table VIII: **(Relative error, ARI, Relative computational time)** - average over groups of number of clusters-.

METHOD	$K \leq 3$	$3 < K < 10$	$K \geq 10$
KMR	$(5.1 \times 10^{-3}, 0.75, 0.18)$	$(1.6 \times 10^{-2}, 0.91, 0.40)$	$(3.0 \times 10^{-2}, 0.70, 0.41)$
PCA	$(1.3 \times 10^{-4}, 0.97, 0.46)$	$(1.6 \times 10^{-3}, 0.97, 0.72)$	$(1.4 \times 10^{-5}, 0.85, 0.47)$
RP	$(1.9 \times 10^{-2}, 0.59, 0.20)$	$(4.5 \times 10^{-2}, 0.80, 0.49)$	$(8.1 \times 10^{-2}, 0.58, 0.41)$
SVD	$(1.0 \times 10^{-5}, 0.97, 0.47)$	$(1.8 \times 10^{-3}, 0.96, 0.64)$	$(2.0 \times 10^{-5}, 0.85, 0.48)$

Table IX: **(Relative error, ARI, Relative computational time)** - average over groups of number of instances-.

METHOD	$n \leq 750$	$750 < n < 2500$	$n \geq 2500$
KMR	$(1.1 \times 10^{-2}, 0.81, 0.28)$	$(5.0 \times 10^{-3}, 0.89, 0.24)$	$(3.0 \times 10^{-2}, 0.62, 0.31)$
PCA	$(2.0 \times 10^{-3}, 0.88, 0.62)$	$(1.6 \times 10^{-5}, 0.97, 0.43)$	$(9.3 \times 10^{-5}, 0.98, 0.48)$
RP	$(3.4 \times 10^{-2}, 0.68, 0.33)$	$(5.6 \times 10^{-2}, 0.65, 0.25)$	$(4.2 \times 10^{-2}, 0.55, 0.33)$
SVD	$(2.7 \times 10^{-4}, 0.87, 0.56)$	$(2.2 \times 10^{-5}, 0.97, 0.46)$	$(1.7 \times 10^{-4}, 0.98, 0.49)$

Regardless of the characteristics of the clustering problem, it can be seen that KMR shows a very competitive performance, in both quality of the obtained solution and computational time, when compared to different feature selection techniques, with particular emphasis on LS. Furthermore, our extensive experimental analysis also shows that, even when KMR is a feature selection technique, it is able to outperform, in both accuracy and computational time, a well-known feature extraction technique used for the K-means problem, such as RP, which makes it a very suitable pre-processing alternative to other approaches that may not scale well on massive data sets, such as SVD and PCA.

V. CONCLUSIONS

In this work, we propose a fully-parellelizable cheap feature selection technique intended for the K-means algorithm called the K-means Relevance Feature Subset Selection algorithm, or just KMR. It consists of applying any λ-approximate K-means heuristic on small subsets of dimensions of the original data set and of using the obtained clustering information to measure the relevance of each feature. In particular, the relevance measure is an upper-bound of the increase in the K-means error that occurs when a certain feature is removed. Among other benefits, this relevance measure is additive and therefore the cost of the proposed method, on each subset of dimensions, is $\mathcal{O}(m \cdot \max\{n \cdot K, \log m\})$, where m is the number of features selected when using Lloyd's algorithm as the λ-approximate K-means heuristic.

In practice, on a wide variety of real life data sets, we compared KMR to different well-known feature selection and feature extraction techniques, commonly used for the K-means problem. The obtained results testify that KMR regularly obtains solutions with lower K-means error than all the considered feature selection techniques: Laplacian scores, Multi-Cluster Feature Selection, maximum variance and random selection. It also requires similar or lower computational times than these approaches. Even more interesting, when compared to feature extraction techniques, such as Random Projections, KMR also shows a noticeable improvement in both error and computational time. We additionally provide different theoretical guarantees in terms of the quality of the approximations obtained via KMR.

As a future step, since our technique consists of solving multiple small-dimensional K-means problems, we would like to analyze the effect of using different coreset techniques, such as [18]–[23], to further reduce the computational requirements of our approach. Furthermore, we plan to design a feature extraction technique of the same nature of KMR. This will allow us not to fully lose the information of a given feature when not selected.

APPENDIX

The Appendix is divided into 4 sections. We first present the proofs of the theorems stated throughout the article, see Appendix A. Afterwards, we present some additional empirical results to analyze the accuracy of the feature selection criteria stated in Algorithm 2, see Appendix B. Moreover, in Appendix C, we extend on the feature selection step commented in Section III and, finally, in Appendix D, we present some additional experiments to further discuss the effect of the number of clusters on the performance of KMR.

A. Proofs

As commented in Section II, the proposed feature selection in Algorithm 3-4 is mainly based on the error bound proposed

in Theorem 1. The definition and theoretical result presented in Definition 1 and Theorem 1 offer a simple way of quantifying the importance of a set of features in terms of their impact on the quality of the obtained clustering. This measurement consists of fixing the corresponding entry on each center of mass to the center of mass of the dimension, and estimating the increase of the error that it implies.

Theorem 1. *Given a data set X with features D, a clustering $\mathcal{P} = \{P_1, \ldots, P_K\}$ and its associated set of centroids, C, then, for any subset of features $S \subseteq D$, the following inequality holds:*

$$E^D(C^S) \leq E^D(C) + r_{D\setminus S} \tag{3}$$

Proof. Firstly, we know that the following inequality holds

$$\sum_{\mathbf{x}\in P_l} \|\mathbf{x} - \mathbf{c}_l^S\|^2 = \sum_{\mathbf{x}\in P_l} \|\mathbf{x} - \mathbf{c}_l\|^2 + |P_l| \cdot \|\mathbf{c}_l - \mathbf{c}_l^S\|^2 \tag{8}$$

and so, due to possible clustering re-assignments, from Eq.8, we deduce the following bound

$$\begin{aligned} E^D(C^S) &\leq E^D(C) + \sum_{l=1}^{K} |P_l| \cdot \|\mathbf{c}_l - \mathbf{c}_l^S\|^2 \\ &= E^D(C) + \sum_{l=1}^{K} |P_l| \cdot \sum_{j\in D\setminus S} (\mathbf{c}_{l,j} - \overline{X^{\{j\}}})^2 \\ &= E^D(C) + \sum_{j\in D\setminus S} r_{\{j\}} \\ &= E^D(C) + r_{D\setminus S} \end{aligned} \tag{9}$$

□

In Corollary 1 and Algorithm 2 in Section II, we describe a simple feature selection process, based on Theorem 1, that leads to a $(1+\epsilon)$- approximation of $E^{D_i}(C_i)$:

Corollary 1. *Given a data set X with features D, a clustering $\mathcal{P}$, its associated set of centroids C, and a subset of features $S \subseteq D$, then C^S is a $(1+\epsilon)$-approximation of $E^D(C)$, where*

$$\epsilon \leq \epsilon_D^S = \frac{r_{D\setminus S}}{E^D(C)} \tag{4}$$

Proof. From Theorem 1, we know that the following inequality holds

$$\begin{aligned} E^D(C^S) &\leq E^D(C) + r_{D\setminus S} \\ &= E^D(C) + \frac{r_{D\setminus S}}{E^D(C)} \cdot E^D(C) \\ &= (1 + \epsilon_D^S) \cdot E^D(C) \end{aligned}$$

□

Proof. The construction proposed in this result can be directly verified from Corollary 1: Since D is sorted increasingly with respect to the relevances $\{r_{\{j\}}\}_{j\in D}$ and l is defined as the largest index for which $r_{D_{1:l}} \leq \epsilon \cdot E^D(C)$ holds, then $I = D_{1:l}$ is the set of dimensions of maximal cardinality, constructed via the relevance measure proposed in Theorem 1, for which

$$\begin{aligned} E^D(C^S) &\leq E^D(C) + \sum_{j\in\ I} r_{\{j\}} \\ &\leq E^D(C) + r_{D_{1:l}} \\ &= E^D(C) + \epsilon_D^S \cdot E^D(C) \\ &\leq (1+\epsilon) \cdot E^D(C), \end{aligned} \tag{10}$$

is satisfied. This is, $S = D \setminus I$ is the subset of dimensions of minimal cardinality that we can select, i.e., that we do not need to fix to a given value to keep a $(1+\epsilon)$- approximation of $E^D(C)$. □

In Theorem 2, we propose a bound to the clustering quality of the approximation obtained via Corollary 1. Such a bound depends on the predefined $\epsilon > 0$, the approximation ratio λ of algorithm $\mathcal{A}$ (which could be the K-means algorithm or any coreset-type approach, for instance) and φ, which, for each party of dimensions, measures the ratio between the 1-means optimal error and sum of the K-means error on the selected variables.

Theorem 2. *Given a data set X with features D, a constant $\epsilon > 0$ and a partition of D, $\{D_1, \ldots, D_t\}$, if S_i is the set of features selected via Algorithm 2 from D_i, for $\epsilon > 0$, then the output of a λ-approximate K-means algorithm (algorithm $\mathcal{A}$) on $S = \bigcup_{i=1}^{t} S_i$, C^*, satisfies*

$$E^D(C^*) \leq \varphi \cdot (1+\epsilon) \cdot E^D(C_{opt}) - \delta \tag{6}$$

where $C_{opt} = \underset{C\subseteq\mathbb{R}^d, |C|=K}{\arg\min} E^D(C)$, $\varphi = \lambda^2 \cdot \frac{\Theta_S}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)}$ with $\Theta_S = E^S(\{\overline{X^S}\})$ and $\delta = \frac{\lambda\cdot(1+c)-t}{t} \cdot \Theta_{D\setminus S}$ with $c = \frac{\sum_{i=1}^{t} E_{D_i}^{S\setminus S_i}(C_i)}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)}$.

Proof. Before proceeding with the proof, we introduce the following notation: given two subsets of dimensions $\tilde{D}, \tilde{\tilde{D}} \subseteq D$, we define $E_{\tilde{D}}^{\tilde{\tilde{D}}}(X) = \sum_{\mathbf{x}\in X} \|\mathbf{x} - \mathbf{c}_{\mathbf{x}}\|_{\tilde{D}}^2$, where $\mathbf{c}_{\mathbf{x}} = \underset{\mathbf{c}\in C}{\arg\min} \|\mathbf{x} - \mathbf{c}\|_{\tilde{\tilde{D}}}^2$. Observe that, for any subset $\widetilde{\tilde{D}} \subseteq D$, $E^{\tilde{D}}(X) = E_{\tilde{D}}^{\tilde{D}}(X) \leq E_{\tilde{\tilde{D}}}^{\tilde{D}}(X)$.

We first apply algorithm $\mathcal{A}$ on each D_i, which generates a set of centroids, C_i, satisfying

$$\begin{aligned} E^{D_i}(C_i^{S_i}) &\leq (1+\epsilon) \cdot E^{D_i}(C_i) \leq \lambda \cdot (1+\epsilon) \cdot E^{D_i}(C_{opt}) \\ &\leq \lambda \cdot (1+\epsilon) \cdot E_D^{D_i}(C_{opt}) \ \forall\ i \in \{1, \ldots, t\} \end{aligned} \tag{11}$$

and so, if we add all the elements in Eq.11, we get

$$\sum_{i=1}^{t} E^{D_i}(C_i^{S_i}) \leq \lambda \cdot (1+\epsilon) \cdot E^D(C_{opt}). \tag{12}$$

Furthermore, observe that

$$\sum_{i=1}^{t} E^{D_i}(C_i^{S_i}) = \underbrace{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i^{S_i})}_{\text{Non-fixed error}} + \underbrace{\Theta_{D\setminus S}}_{\text{Fixed error}} \tag{13}$$

From now on, we focus on bounding the "Non-fixed error" (T). We first apply algorithm $\mathcal{A}$ on $S = \bigcup_{i=1}^{t} S_i$ and obtain a set of K centroids, C^*:

$$\begin{aligned} E^S(C^*) &\leq \lambda \cdot E^S(C_i^{S_i}) \leq \lambda \cdot E_{D_i}^S(C_i^{S_i}) \\ &= \lambda \cdot [E_{D_i}^{S_i}(C_i^{S_i}) + E_{D_i}^{S\setminus S_i}(C_i^{S_i})] \;\; \forall\, i \in \{1,\ldots,t\}, \end{aligned}$$

and so,

$$E^S(C^*) \leq \frac{\lambda}{t} \cdot [T + \sum_{i=1}^{t} E_{D_i}^{S\setminus S_i}(C_i^{S_i})]. \tag{14}$$

If we denote $S = \sum_{i=1}^{t} E_{D_i}^{S\setminus S_i}(C_i^{S_i})$, we would like to determine an upper bound to c such that $S = c \cdot T$. This bound is of relevance since, considering Eq.12 and Eq.14, we know that

$$\begin{aligned} E^D(C^*) &\leq \frac{\lambda^2}{t} \cdot (1+\epsilon) \cdot (1+c) \cdot E^D(C_{opt}) \\ &- \frac{\lambda \cdot (1+c) - t}{t} \cdot \Theta_{D\setminus S} \end{aligned} \tag{15}$$

We now focus on the factor $c = \frac{\sum_{i=1}^{t} E_{D_i}^{S\setminus S_i}(C_i^{S_i})}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i^{S_i})} = \frac{\sum_{i=1}^{t}\sum_{j\neq i} E_{D_i}^{S_j}(C_i)}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)} \leq \frac{(t-1)\cdot\Theta_S}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)}$. Furthermore,

$$\frac{(\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)) + (t-1)\cdot\Theta_S}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)} \leq \frac{t\cdot\Theta_S}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)} \tag{16}$$

Hence, using Eq.16, we can bound Eq.15 using information that we know in advance, as follows

$$\begin{aligned} E^D(C^*) &\leq \lambda^2 \cdot (1+\epsilon) \cdot \frac{\Theta_S}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)} \cdot E^D(C_{opt}) - \delta \\ &= \lambda^2 \cdot (1+\epsilon) \cdot \frac{\Theta_S}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)} \cdot E^D(C_{opt}) - \delta \\ &= \varphi \cdot (1+\epsilon) \cdot E^D(C_{opt}) - \delta \end{aligned}$$

□

Observation 1. *The value of δ in Eq.6 (Theorem 2) is commonly non-negative.*

Proof. To observe this, we could, for instance, impose the condition $E_{D_i}^{S_i}(C_i) \leq E_{D_j}^{S_i}(C_j)$, for all $j \neq i$. If this condition is not satisfied, we have the following chain of inequalities

$$\begin{aligned} E^{D_i}(C_j^{S_i}) &\leq E_{D_j}^{D_i}(C_j^{S_i}) = E_{D_j}^{S_i}(C_j) + \Theta_{D_i\setminus S_i} \\ &\leq E_{D_i}^{S_i}(C_i) + \Theta_{D_i\setminus S_i} = E^{D_i}(C_i^{S_i}) \\ &\leq (1+\epsilon)\cdot E^{D_i}(C_j) \end{aligned} \tag{17}$$

This is, $C_j^{S_i}$ provides a $(1+\epsilon)$-approximation of C_i on the set of dimensions D_i, even when it is obtained on the subset of dimensions D_j. When this condition is satisifed, we have

$$\begin{aligned} c &= \frac{\sum_{i=1}^{t} E_{D_i}^{S\setminus S_i}(C_i)}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)} = \frac{\sum_{i=1}^{t}\sum_{j\neq i} E_{D_i}^{S_j}(C_i)}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)} \\ &= \frac{\sum_{j=1}^{t-1}[(\sum_{i=j+1}^{t} E_{D_j}^{S_i}(C_j)) + (\sum_{i=1}^{j} E_{D_{j+1}}^{S_i}(C_{j+1}))]}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)} \\ &\geq \frac{\sum_{j=1}^{t-1}\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)}{\sum_{i=1}^{t} E_{D_i}^{S_i}(C_i)} \geq t-1 \end{aligned} \tag{18}$$

Therefore, $\delta \geq (\lambda - 1)\cdot\Theta_{D\setminus S} \geq 0$.

□

The following result is a corollary of Theorem 2 and it is used in Algorithm 4 as as criterion for selecting the variables.

Corollary 2. *If $S = \bigcup_{i=1}^{t} S_i^*$ is the set of m features selected by Algorithm 3 on a data set X with features D, then the output of a λ-approximate K-means algorithm (algorithm $\mathcal{A}$) on S, C^*, satisfies*

$$E^D(C^*) \leq \varphi \cdot (1+\epsilon^*) \cdot E^D(C_{opt}) - \delta, \tag{7}$$

where $\epsilon^* = \max_{i\in\{1,\ldots,t\}} \epsilon_{D_i}^{S_i^*}$.

Proof. From Algorithm 2, the following inequality chain holds for all $i \in \{1,\ldots,t\}$

$$E^{D_i}(C_i^{S_i}) \leq (1+\epsilon_{D_i}^{S_i}) \cdot E^{D_i}(C_i) \leq (1 + \max_{j\in\{1,\ldots,t\}} \epsilon_{D_j}^{S_j}) \cdot E^{D_i}(C_i)$$

The rest of the proof is analogous to that of Theorem 2. □

B. Some experimental results using Algorithm 2

In this section, we provide some additional experimental results using the feature selection strategy proposed in Algorithm 2. In particular, we consider all 16 data sets used in Section IV and apply the feature selection approach discussed in Algorithm 2 to obtain a $(1+\epsilon)$-approximation of the solution obtained via KM++, for $\epsilon \in \{0.01, 0.05, 0.10, 0.50, 1.01\}$.

As commented in Section III, Algorithm 2 is primarily based on the variable importance score proposed in Theorem 1. Such a score is a bound to the error increase that would take place if all the centers of mass obtained by algorithm $\mathcal{A}$ are fixed on a predefined subset of dimensions. This bound does not take into consideration possible reassignements of clusters which notoriously fasten the selection procedure, as there is no need to compute an additional cluster reassignment step, which would be $\mathcal{O}(n\cdot K\cdot m)$, and, more importantly, the effect of each variable is additive, meaning that the corresponding

error increase of each dimension is independent of the subset of fixed dimensions considered, i.e., we do not need to evaluate all the possible combinations of fixed dimensions separately.

In Fig. 4, we show the obtained epsilon (which must be $\leq \epsilon$) obtained by Algorithm 2. Afterwards, we present the error (difference) between the epsilon predicted by Algorithm 2 and the epsilon obtained for the same features selected by Algorithm 2, computing all clustering reassignements, see Fig. 5. Finally, in Fig. 6, we observe the number of features discarded (fixed) to reach the $(1+\epsilon)$-approximation.

Table X: Average, over all data sets, for the results presented in Fig. 4-6.

ϵ	ALGORITHM 2 EPSILON	ERROR	DISCARDED VARIABLES
0.01	8.03×10^{-3}	1.76×10^{-4}	0.70
0.05	4.09×10^{-2}	8.85×10^{-4}	0.79
0.10	8.26×10^{-2}	3.18×10^{-3}	0.83
0.50	3.02×10^{-1}	1.53×10^{-2}	0.93
1.01	5.13×10^{-1}	5.33×10^{-2}	0.96

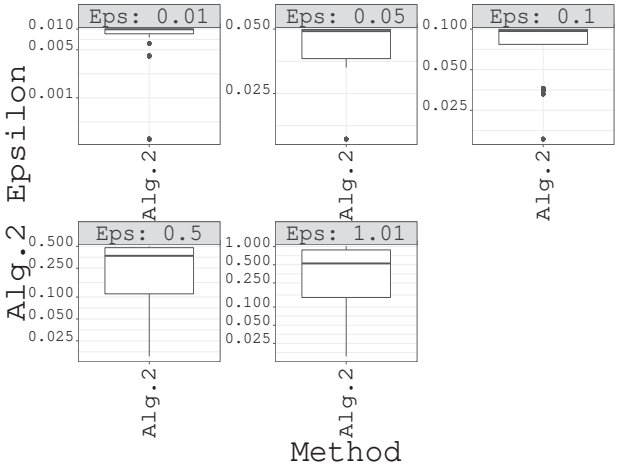

Fig. 4. Epsilon obtained after applying Algorithm 2.

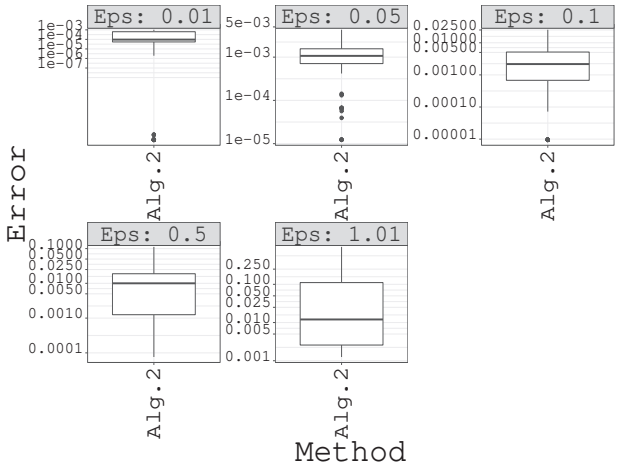

Fig. 5. Error obtained after selecting the last variable for which Algorithm 2 achieves the $(1+\epsilon)$-approximation.

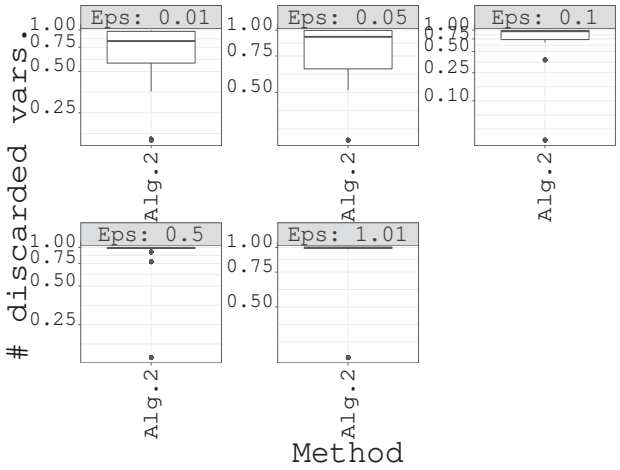

Fig. 6. Proportion of variables discarded by Algorithm 2 for reaching the $(1+\epsilon)$-approximation.

According to Fig. 4, we observe that, as the value of ϵ decreases, the epsilon obtained by Algorithm 2 is closer to it. This observation is partially related to the fact that, as we increase ϵ, and, therefore, discard more variables, we are mostly left with those variables with the highest error increments, reason for which adding an extra variable to those features that are already discarded may easily exceed the $(1+\epsilon)$- bound. In this scenario (largest values of ϵ), we observe that the error of the approximation provided by Theorem 1 loses more accuracy, as in this case we are fixing more variables and, therefore, adding more error to the bound. However, we must point out that the error obtained is still negligible with respect to the ϵ chosen: In Fig. 5 and Tab.X, we observe that, in large majority of the cases, such an error is under 5% of the actual value of ϵ.

Regardless of the value selected for ϵ, Algorithm 2 is always able to discard a large amount of the variables, while generating a $1+\epsilon$-approximation of the best solution obtained via KM++. In particular, for $\epsilon = 0.01$, Algorithm 2 eliminated, on average, 70% of the variables to reach the expected quality and, for $\epsilon = 0.50$, over 90% of the variables are already fixed and the error bound is satisfied.

C. Feature Selection of KMR

As commented in Section III, in Algorithm 3, after computing the relevances of each feature on each chunk D_i, the features are sorted increasingly and all the partial sums of the associated relevances are computed, $\{\xi_{D_i}^{|D_i|}, \xi_{D_i}^{|D_i|-1}, \ldots, \xi_{D_i}^{0}\}$ for all $i \in \{1, \ldots, t\}$, where $\xi_{D_i}^{j}$ stands for the epsilon associated to the error increase that occurs when only selecting the j most relevant variables in D_i.

The feature selection problem, Eq.5, could be easily solved by selecting the m largest entries among the t lists. However, all these lists have been previously sorted in Algorithm 2, hence we take advantage of this feature to propose a simple $\mathcal{O}(t)$ time per iteration heuristic to solve Eq.5:

The heuristic proposed in Algorithm 4 is fairly simple as it just needs to update the maximum of $\{\xi_{D_1}^{d_1}, \xi_{D_2}^{d_2}, \ldots, \xi_{D_t}^{d_t}\}$

Algorithm 4: Feature Selection

Input: $\{\xi_{D_i}^{|D_i|}, \xi_{D_i}^{|D_i|-1}, \ldots, \xi_{D_i}^{0}\}$ for all $i \in \{1, \ldots, t\}$.
Output: Set of m features selected, $S \subseteq \{1, \ldots, d\}$.

- Take a set of non-negative integers $\{d_1, \ldots, d_t\}$, satisfying $\sum_{i=1}^{t} d_i = m$ and set $\xi_{D_i}^{-1} \to \infty$, for all $i \in \{1, \ldots, t\}$.
while $\max_{i \in \{1,\ldots,t\}} \xi_{D_i}^{d_i} > \min_{i \in \{1,\ldots,t\}} \xi_{D_i}^{d_i - 1}$ **do**
  - Set $d_j = d_j + 1$, where $j = \arg\max_{i \in \{1,\ldots,t\}} \xi_{d_i}^{i}$.
  - Set $d_l = d_l - 1$, where $l = \arg\min_{i \in \{1,\ldots,t\}} \xi_{d_i - 1}^{i}$.
end
- For all $i \in \{1, \ldots, t\}$, set S_i as the most relevant d_i features in D_i.
Return $S = \bigcup_{i=1}^{t} S_i$.

and the minimum of $\{\xi_{D_1}^{d_1-1}, \xi_{D_2}^{d_2-1}, \ldots, \xi_{D_t}^{d_t-1}\}$, which implies, at most, a $\mathcal{O}(t)$ time cost. Since the arrays $\{\xi_{D_i}^{|D_i|}, \xi_{D_i}^{|D_i|-1}, \ldots, \xi_{D_i}^{0}\}$ are all sorted for $i \in \{1, \ldots, t\}$, then when the condition $\max_{i \in \{1,\ldots,t\}} \xi_{D_i}^{d_i} \leq \min_{i \in \{1,\ldots,t\}} \xi_{D_i}^{d_i - 1}$, is satisfied, $\max_{i \in \{1,\ldots,t\}} \xi_{D_i}^{d_i}$ is minimized.

D. Additional experiments: The effect of the number of clusters

In this section, we comment on some complimentary experimental results to those presented in Section IV. Our goal is to observe the effect of the number of clusters on the performance of KMR with respect to the feature selection/extraction methods used in Section IV. In particular, we will analyze the performance of these algorithms on a variety of real data sets, see Table XII, and a multiple number of clusters, $K \in \{2, 3, 5, 10, 25, 50, 100\}$, for each case. Furthermore, we have fixed the number of variables to be selected/extracted to $m = 10$. Due to the random nature of the experimental setting, each experiment has been repeated 20 times.

Table XII: Information of the data sets.

Data Set	n	d
Handwritten Numerals	2000	216
Parkinson	5875	22
Cambridge First Order	6118	58
WiFi Fingerprint	19937	529
Superconductivity	21263	82
Buzz Social Media	28179	97

In Table XI, we show the averages of the relative K-means error, the ARI with respect to the clustering obtained via KM++ and the relative computational time for the considered methods over all data sets. Furthermore, due to lack of space, in the supplementary figures shown in https://github.com/MarcoVCapo/KMR_AppendixD, we present the boxplots of all these factors divided between feature selection and feature extraction algorithms.

Similarly to the results presented in Section IV, we can observe that KMR is consistently the feature selection approach that leads to the most competitive approximations (lowest relative error and highest ARI with respect to KM++) to the corresponding K-means problem regardless of the number of clusters considered. Moreover, it should also be remarked that LS generated, on average, approximations with the same order of magnitude of relative error as KMR, however it required a much longer computational time than KMR: For instance, for $K = 2$, it required, on average, 650.55 times more computational time than that required by KMR. Furthermore, both MCFS and Rand failed to generate competitive approximations regardless of the number of clusters considered.

In terms of the feature extraction methods, we observe again that KMR is able to outperform RP in both computational time and quality of the approximation. However, as the number of clusters is increased, both PCA and SVD tend to converge faster than KMR as their time complexities are independent of this factor. Also, in terms of the quality of the obtained solution, KMR, for a large number of clusters, might not be

Table XI: **(Relative error, ARI, Relative computational time)** - average over groups of number of clusters-.

METHOD	$K = 2$	$K = 3$	$K = 5$	$K = 10$
KMR	$(4.8 \times 10^{-3}, 0.93, 0.36)$	$(1.9 \times 10^{-2}, 0.85, 0.38)$	$(4.4 \times 10^{-2}, 0.79, 0.39)$	$(6.3 \times 10^{-2}, 0.76, 0.44)$
LS	$(5.9 \times 10^{-3}, 0.91, 234.20)$	$(3.2 \times 10^{-2}, 0.79, 149.43)$	$(4.6 \times 10^{-2}, 0.77, 79.01)$	$(1.1 \times 10^{-2}, 0.72, 36.92)$
MAXVAR	$(2.7 \times 10^{-2}, 0.77, 0.36)$	$(5.1 \times 10^{-2}, 0.73, 0.39)$	$(8.0 \times 10^{-2}, 0.76, 0.38)$	$(9.4 \times 10^{-2}, 0.76, 0.42)$
MCFS	$(1.4 \times 10^{0}, 0.23, 30.61)$	$(1.0 \times 10^{1}, 0.31, 20.42)$	$(2.9 \times 10^{3}, 0.31, 12.0)$	$(1.2 \times 10^{4}, 0.35, 6.6)$
RAND	$(1.2 \times 10^{0}, 0.38, 0.39)$	$(9.3 \times 10^{0}, 0.42, 0.36)$	$(2.8 \times 10^{3}, 0.34, 0.36)$	$(1.2 \times 10^{4}, 0.39, 0.40)$
PCA	$(3.5 \times 10^{-6}, 0.99, 0.56)$	$(1.7 \times 10^{-8}, 0.95, 0.51)$	$(2.3 \times 10^{-5}, 0.93, 0.48)$	$(4.9 \times 10^{-6}, 0.87, 0.45)$
RP	$(2.8 \times 10^{-2}, 0.78, 0.38)$	$(3.6 \times 10^{-2}, 0.77, 0.39)$	$(6.2 \times 10^{-2}, 0.75, 0.44)$	$(1.0 \times 10^{-1}, 0.72, 0.55)$
SVD	$(3.1 \times 10^{-6}, 0.99, 0.55)$	$(7.1 \times 10^{-9}, 0.94, 0.49)$	$(8.6 \times 10^{-7}, 0.92, 0.46)$	$(4.8 \times 10^{-7}, 0.88, 0.46)$

METHOD	$K = 25$	$K = 50$	$K = 100$
KMR	$(1.5 \times 10^{-1}, 0.65, 0.48)$	$(2.2 \times 10^{-1}, 0.60, 0.52)$	$(3.3 \times 10^{-1}, 0.56, 0.54)$
LS	$(2.6 \times 10^{-1}, 0.62, 16.12)$	$(4.1 \times 10^{-1}, 0.56, 9.04)$	$(6.1 \times 10^{-1}, 0.51, 4.63)$
MAXVAR	$(2.0 \times 10^{-1}, 0.62, 0.46)$	$(2.8 \times 10^{-1}, 0.58, 0.48)$	$(3.9 \times 10^{-1}, 0.53, 0.49)$
MCFS	$(6.3 \times 10^{4}, 0.34, 3.33)$	$(9.5 \times 10^{4}, 0.34, 2.11)$	$(1.3 \times 10^{5}, 0.31, 1.42)$
RAND	$(6.7 \times 10^{4}, 0.33, 0.46)$	$(9.4 \times 10^{4}, 0.27, 0.49)$	$(1.2 \times 10^{5}, 0.27, 0.52)$
PCA	$(3.9 \times 10^{-3}, 0.76, 0.45)$	$(1.5 \times 10^{-2}, 0.69, 0.51)$	$(2.8 \times 10^{-2}, 0.65, 0.49)$
RP	$(1.7 \times 10^{-1}, 0.62, 0.55)$	$(2.2 \times 10^{-1}, 0.57, 0.54)$	$(2.5 \times 10^{-1}, 0.57, 0.53)$
SVD	$(5.2 \times 10^{-3}, 0.74, 0.46)$	$(1.8 \times 10^{-2}, 0.69, 0.49)$	$(3.0 \times 10^{-2}, 0.64, 0.49)$

as competitive as the feature extraction methods considered, which is expected as shown in both Theorem 2 and Corollary 2, where the accuracy constant φ increases as the number of clusters is also higher. In this sense, it is clear that, when compared to the feature extraction algorithms, the optimal setting for KMR occurs when the number of clusters considered is not high.

ACKNOWLEDGMENT

Marco Capó and Aritz Pérez are partially supported by the Basque Government through the BERC 2014-2017 program and the ELKARTEK program, and by the Spanish Ministry of Economy and Competitiveness MINECO: BCAM Severo Ochoa excelence accreditation SVP-2014-068574 and SEV-2017-0718, and through the project TIN2017-82626-R funded by (AEI/FEDER, UE). Jose A. Lozano is partially supported by the Basque Government (IT1244-19), and Spanish Ministry of Economy and Competitiveness MINECO (BCAM Severo Ochoa excellence accreditation SEV-2017-0718 and TIN2016-78365-R).

REFERENCES

[1] C. Ding and X. He, "K-means clustering via principal component analysis," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 29.

[2] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[3] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[4] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[5] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip *et al.*, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[6] S. Äyrämö and T. Kärkkäinen, "Introduction to partitioning-based clustering methods with a robust example," *Reports of the Department of Mathematical Information Technology. Series C, Software engineering and computational intelligence 1/2006*, 2006.

[7] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "Np-hardness of Euclidean sum-of-squares clustering," *Machine Learning*, vol. 75, no. 2, pp. 245–248, 2009.

[8] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," in *International Workshop on Algorithms and Computation*, 2009, pp. 274–285.

[9] L. Kaufman and P. Rousseeuw, *Clustering by means of medoids*. North-Holland, 1987.

[10] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[11] L. Bottou and Y. Bengio, "Convergence properties of the k-means algorithms," in *Advances in Neural Information Processing Systems*, 1995, pp. 585–592.

[12] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *IEEE International Conference on Cloud Computing*, 2009, pp. 674–679.

[13] J. Drake and G. Hamerly, "Accelerated k-means with adaptive distance bounds," in *5th NIPS Workshop on Optimization for Machine Learning*, 2012, pp. 42–53.

[14] C. Elkan, "Using the triangle inequality to accelerate k-means," in *Proceedings of the 20th International Conference on Machine Learning*, 2003, pp. 147–153.

[15] G. Hamerly, "Making k-means even faster," in *Proceedings of the 2010 SIAM International Conference on Data Mining*, 2010, pp. 130–140.

[16] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th International conference on World Wide Web*, 2010, pp. 1177–1178.

[17] M. Capó, A. Pérez, and J. A. Lozano, "An efficient k-means clustering algorithm for tall data," *Data Mining and Knowledge Discovery*, pp. 1–36, 2020.

[18] O. Bachem, M. Lucic, and A. Krause, "Scalable and distributed clustering via lightweight coresets," *arXiv preprint arXiv:1702.08248*, 2017.

[19] M.-F. F. Balcan, S. Ehrlich, and Y. Liang, "Distributed k-means and k-median clustering on general topologies," in *Advances in Neural Information Processing Systems*, 2013, pp. 1995–2003.

[20] M. Capó, A. Pérez, and J. A. Lozano, "An efficient approximation to the k-means clustering for massive data," *Knowledge-Based Systems*, vol. 117, pp. 56–69, 2017.

[21] D. Feldman, M. Monemizadeh, and C. Sohler, "A ptas for k-means clustering based on weak coresets," in *Proceedings of the twenty-third annual symposium on Computational geometry*, 2007, pp. 11–18.

[22] S. Har-Peled and S. Mazumdar, "On coresets for k-means and k-median clustering," in *Proceedings of the 36th ACM Symposium on Theory of Computing*, 2004, pp. 291–300.

[23] A. Kumar, Y. Sabharwal, and S. Sen, "A simple linear time $(1+\varepsilon)$-approximation algorithm for k-means clustering in any dimensions," in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, 2004, pp. 454–462.

[24] J. M. Peña, J. A. Lozano, and P. Larranaga, "An empirical comparison of four initialization methods for the k-means algorithm," *Pattern Recognition Letters*, vol. 20, no. 10, pp. 1027–1040, 1999.

[25] A. Vattani, "K-means requires exponentially many iterations even in the plane," *Discrete & Computational Geometry*, vol. 45, no. 4, pp. 596–616, 2011.

[26] S. J. Redmond and C. Heneghan, "A method for initialising the k-means clustering algorithm using kd-trees," *Pattern Recognition Letters*, vol. 28, no. 8, pp. 965–973, 2007.

[27] D. Steinley and M. J. Brusco, "Initializing k-means batch clustering: A critical evaluation of several techniques," *Journal of Classification*, vol. 24, no. 1, pp. 99–121, 2007.

[28] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 1027–1035.

[29] J. Fan, R. Samworth, and Y. Wu, "Ultrahigh dimensional feature selection: beyond the linear model," *Journal of machine learning research*, vol. 10, no. Sep, pp. 2013–2038, 2009.

[30] S. Greenhill and S. Venkatesh, "Distributed query processing for mobile surveillance," in *Proceedings of the 15th ACM international conference on Multimedia*. ACM, 2007, pp. 413–422.

[31] S. Mitra, M. Agrawal, A. Yadav, N. Carlsson, D. Eager, and A. Mahanti, "Characterizing web-based video sharing workloads," *ACM Transactions on the Web (TWEB)*, vol. 5, no. 2, p. 8, 2011.

[32] C. Boutsidis, P. Drineas, and M. W. Mahoney, "Unsupervised feature selection for the k-means clustering problem," in *Advances in Neural Information Processing Systems*, 2009, pp. 153–161.

[33] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu, "Dimensionality reduction for k-means clustering and low rank approximation," in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. ACM, 2015, pp. 163–172.

[34] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the nips 2003 feature selection challenge," in *Advances in neural information processing systems*, 2005, pp. 545–552.

[35] K. Mugunthadevi, S. Punitha, M. Punithavalli, and K. Mugunthadevi, "Survey on feature selection in document clustering," *International Journal on Computer Science and Engineering*, vol. 3, no. 3, pp. 1240–1244, 2011.

[36] D. M. Witten and R. Tibshirani, "A framework for feature selection in clustering," *Journal of the American Statistical Association*, vol. 105, no. 490, pp. 713–726, 2010.

[37] Y. Liang, M.-F. Balcan, and V. Kanchanapally, "Distributed pca and k-means clustering," in *The Big Learning Workshop at NIPS*, vol. 2013. Citeseer, 2013.

[38] M. Holmes, A. Gray, and C. Isbell, "Fast svd for large-scale matrices," in *Workshop on Efficient Machine Learning at NIPS*, vol. 58, 2007, pp. 249–252.

[39] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 245–250.

[40] C. Boutsidis, A. Zouzias, and P. Drineas, "Random projections for k-means clustering," in *Advances in Neural Information Processing Systems*, 2010, pp. 298–306.

[41] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[42] H. Liu and H. Motoda, *Feature selection for knowledge discovery and data mining*. Springer Science & Business Media, 2012, vol. 454.
[43] S. Alelyani, J. Tang, and H. Liu, "Feature selection for clustering: A review," in *Data Clustering*. Chapman and Hall/CRC, 2018, pp. 29–60.
[44] D. Cai, C. Zhang, and X. He, "Unsupervised feature selection for multi-cluster data," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 333–342.
[45] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Advances in neural information processing systems*, 2006, pp. 507–514.
[46] T. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.
[47] G. Louppe and P. Geurts, "Ensembles on random patches," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 346–361.
[48] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "A local search approximation algorithm for k-means clustering," in *Proceedings of the 18th annual Symposium on Computational Geometry*, 2002, pp. 10–18.
[49] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.
[50] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

Marco Capó received his Ph.D. degree in 2019 from the University of the Basque Country and, in 2015, the MS.c. degree in Mathematical Modeling Engineering from Universität Hamburg. He is currently a postdoctoral researcher at the Basque Center for Applied Mathematics. His research interests are in machine learning and optimization, with a particular focus on unsupervised learning problems.

Aritz Pérez received his Ph.D. degree in 2010 from the University of Basque Country, department of Computer Science and Artificial Intelligence. Currently, he is a postdoctoral researcher at the Basque Center for Applied Mathematics. His current scientific interests include supervised, unsupervised and weak classification, probabilistic graphical models, model selection and evaluation, time series and crowd learning.

Jose A. Lozano received his Ph.D. degree in 1998 from the University of the Basque Country. He became a full professor at the Department of Computer Science and Artificial Intelligence in 2008. Since 2005 he has led the Intelligent Systems Group (ISG) based in the Computer Science School. His research areas are evolutionary computation, machine learning and probabilistic graphical models. He has published 4 books, more than 100 scientific ISI journal articles and about 150 contributions to national and international conferences. These publications have received more than 12,400 citations. Prof. Lozano is an associate editor of IEEE Trans. on Evolutionary Computation and IEEE Trans. on Neural Network and Learning Systems among other prestigious journals.